Course Updates

- Rudimentary course webpage is available from:

  http://www.cs.toronto.edu/~ls/

  (look under teaching)

- Lecture notes, slides from last time and Assignment 2 are now posted

- Starter code for programming portion of Assignment 2 will be available in a day or two

Camera Models
Part 2

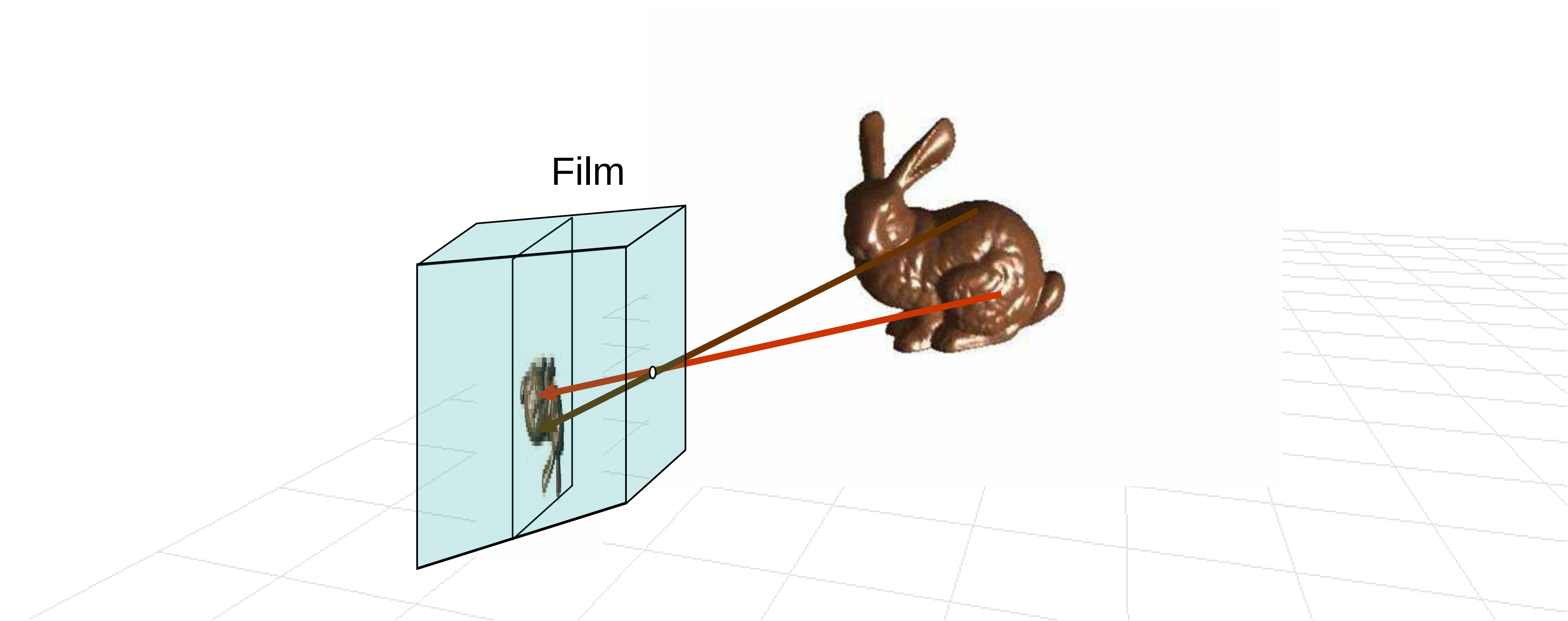
Computer Graphics, CSCD18

Fall 2007

Instructor: Leonid Sigal

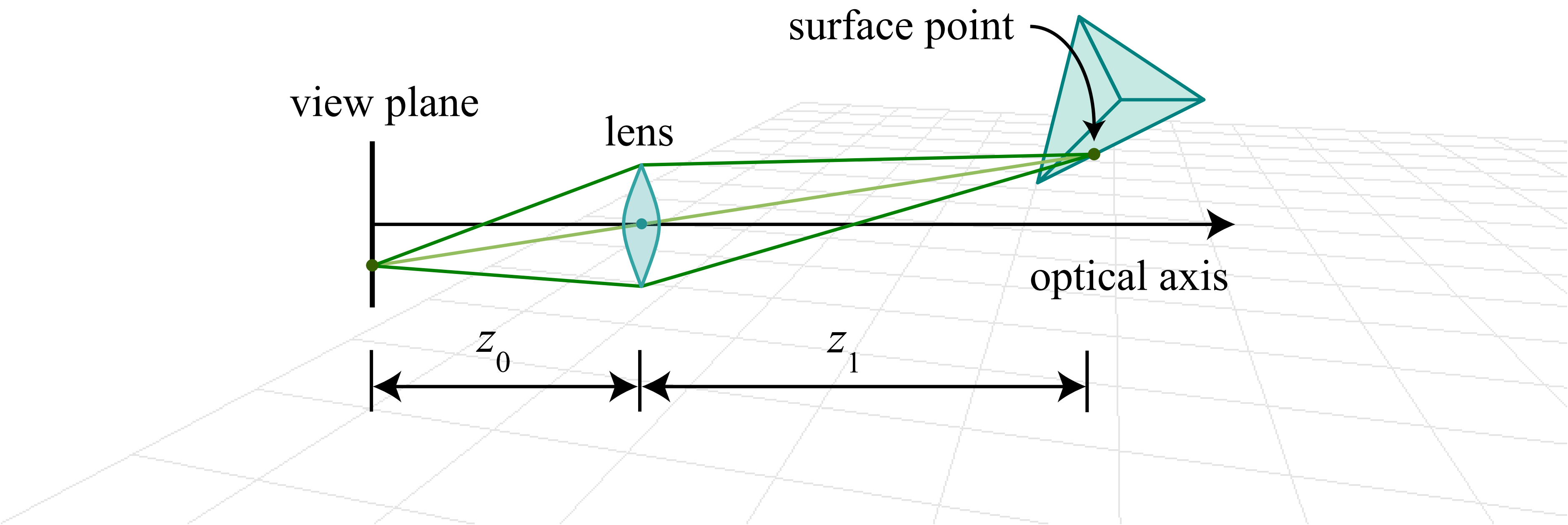
Last time …

- 3D transformations
- Camera models
  - Pinhole camera



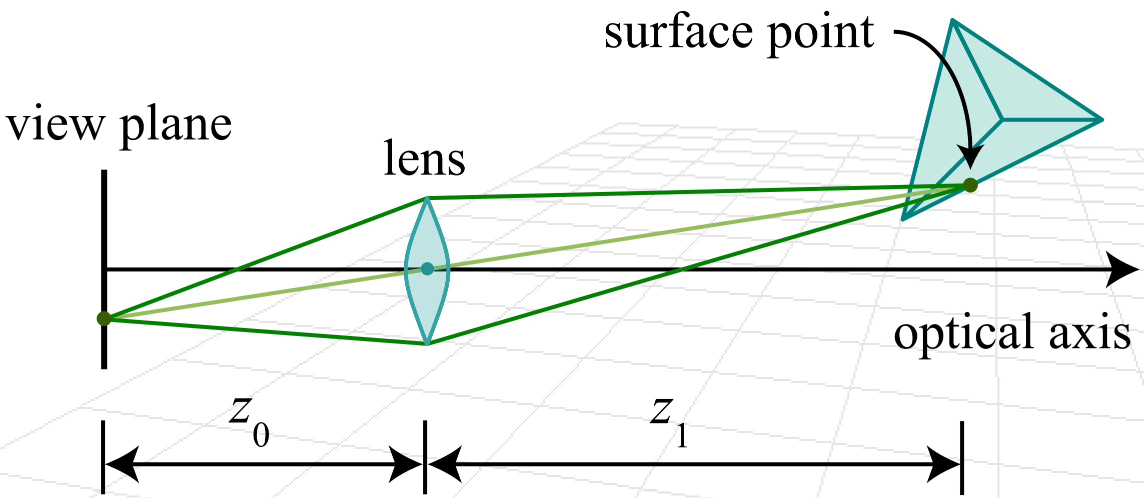
Last time …

- 3D transformations
- Camera models
  - Pinhole camera
  - Thin lens model



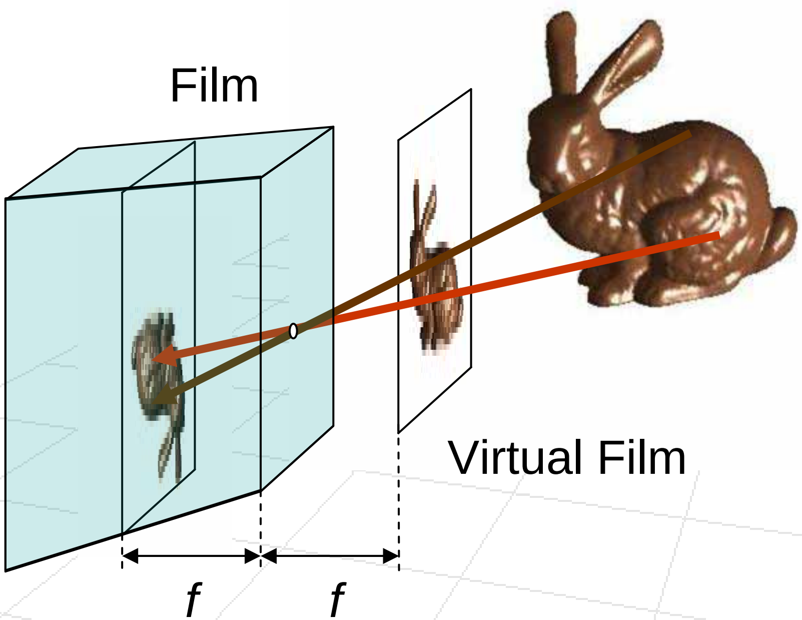
Last time …

- 3D transformations
- Camera models
  - Pinhole camera
  - Thin lens model
  - Relationship between pinhole camera and thin lens model



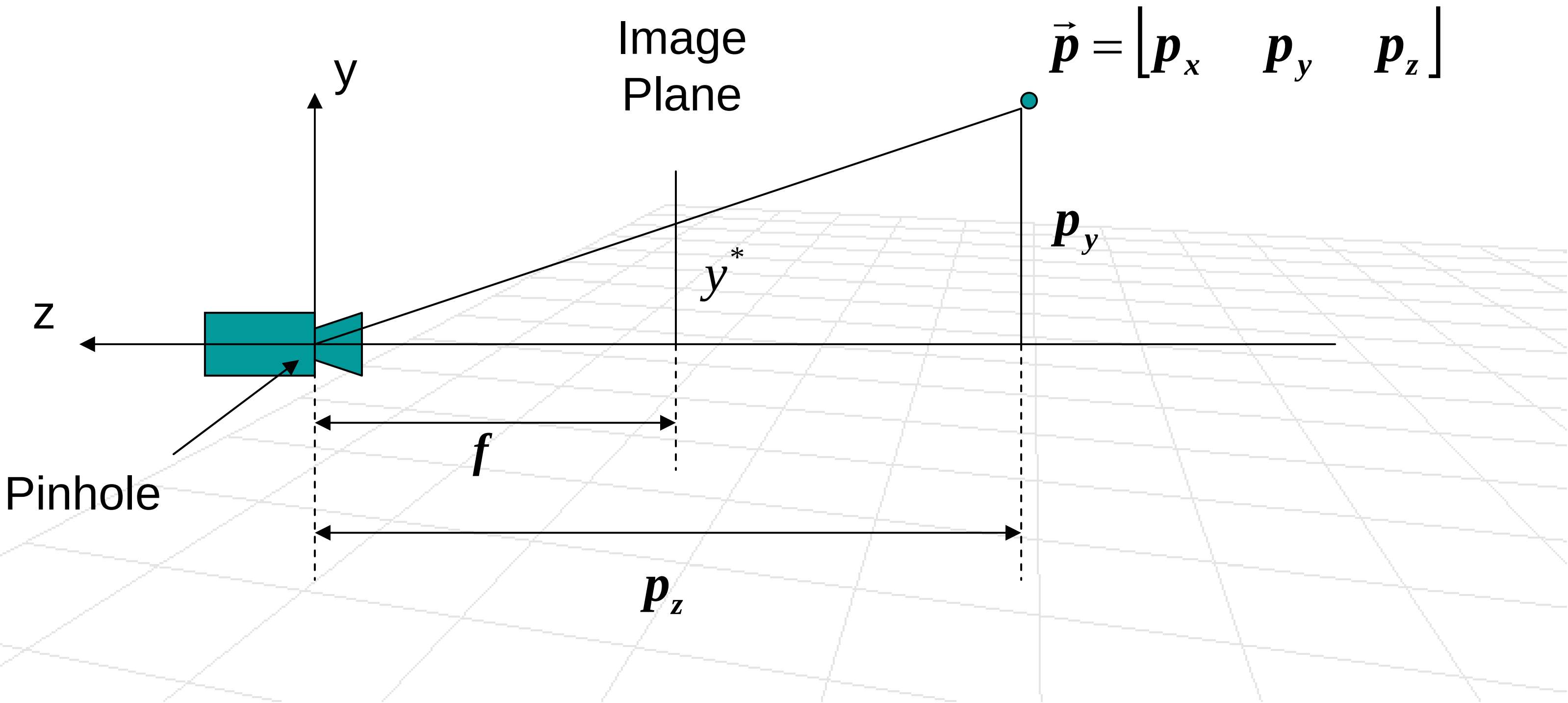
Last time …

- 3D transformations
- Camera models
  - Pinhole camera
  - Thin lens model
  - Relationship between pinhole camera and thin lens model
- Conceptual pinhole camera



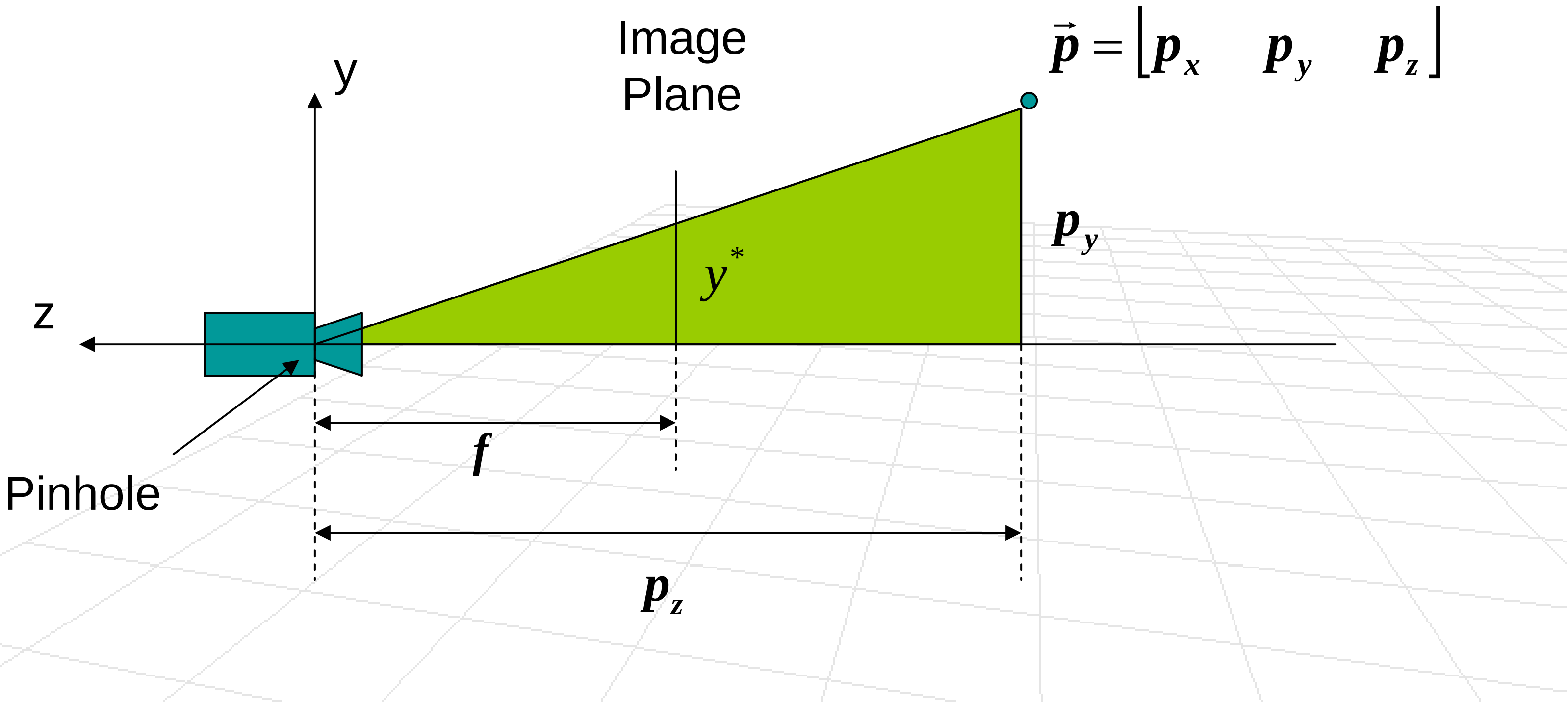
Perspective Projection

- Using similar triangles:



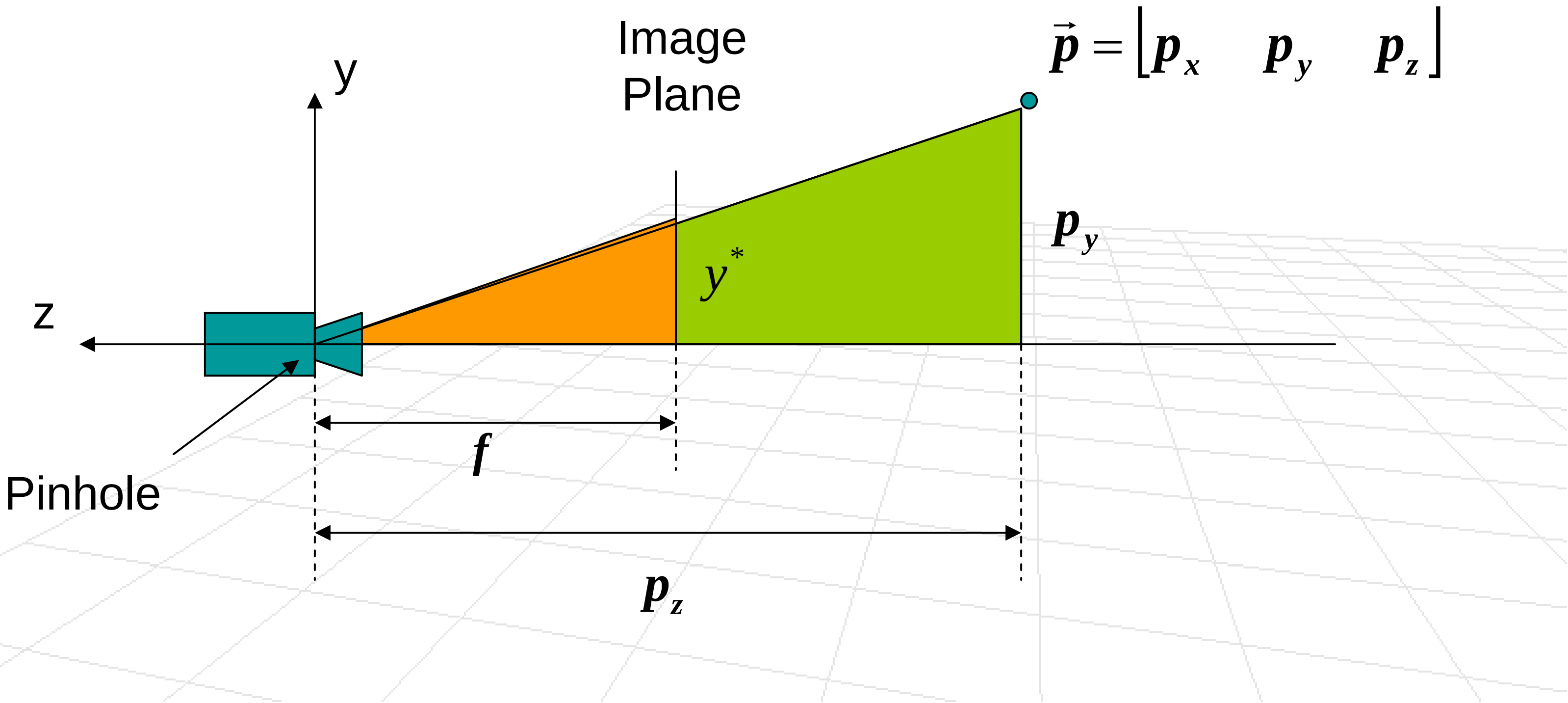
Perspective Projection

- Using similar triangles:



Perspective Projection

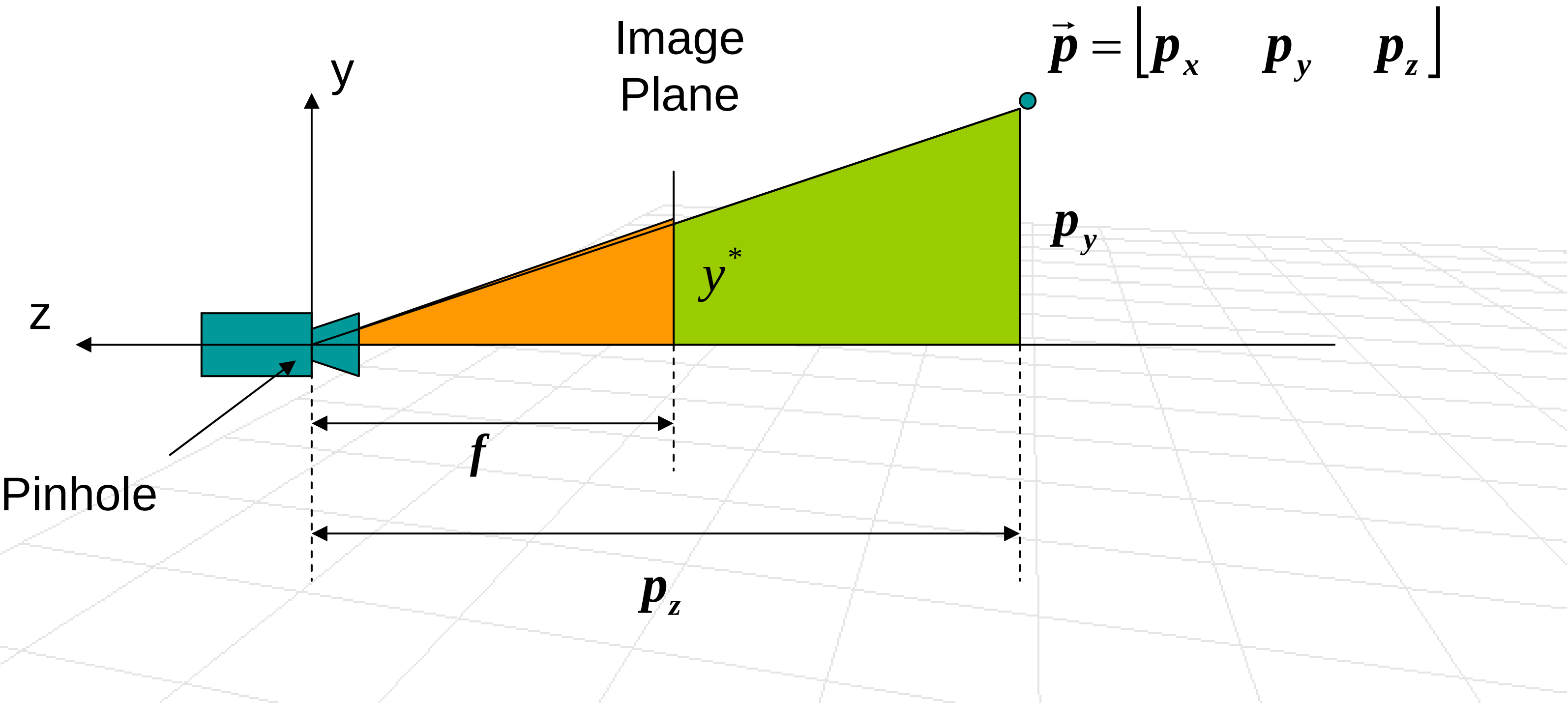
- Using similar triangles:



Perspective Projection

- Using similar triangles: $\frac{y^*}{p_y} = \frac{f}{p_z}$

$$y^* = \frac{f}{p_z} p_y$$

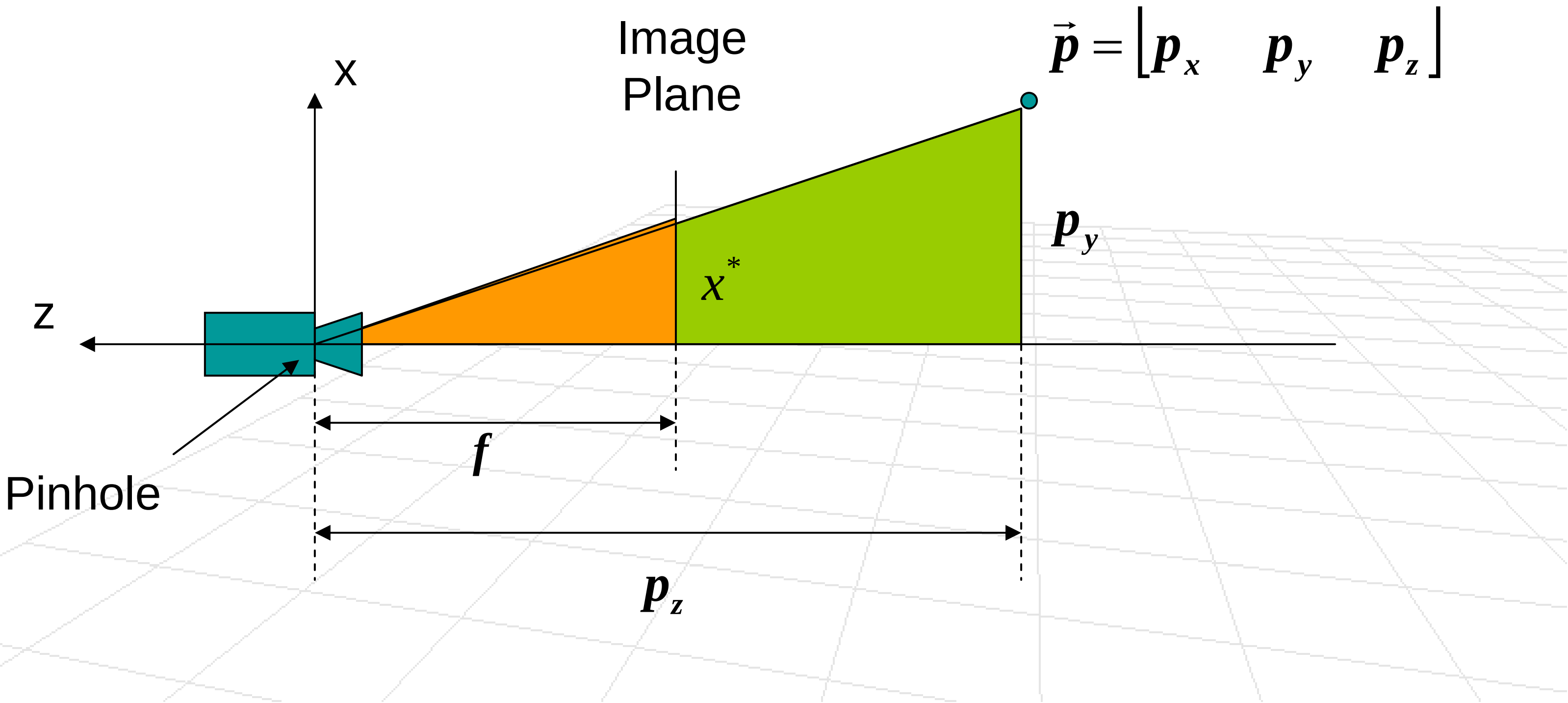


Perspective Projection

- Using similar triangles: $\frac{y^*}{p_y} = \frac{f}{p_z}$ $\frac{x^*}{p_x} = \frac{f}{p_z}$

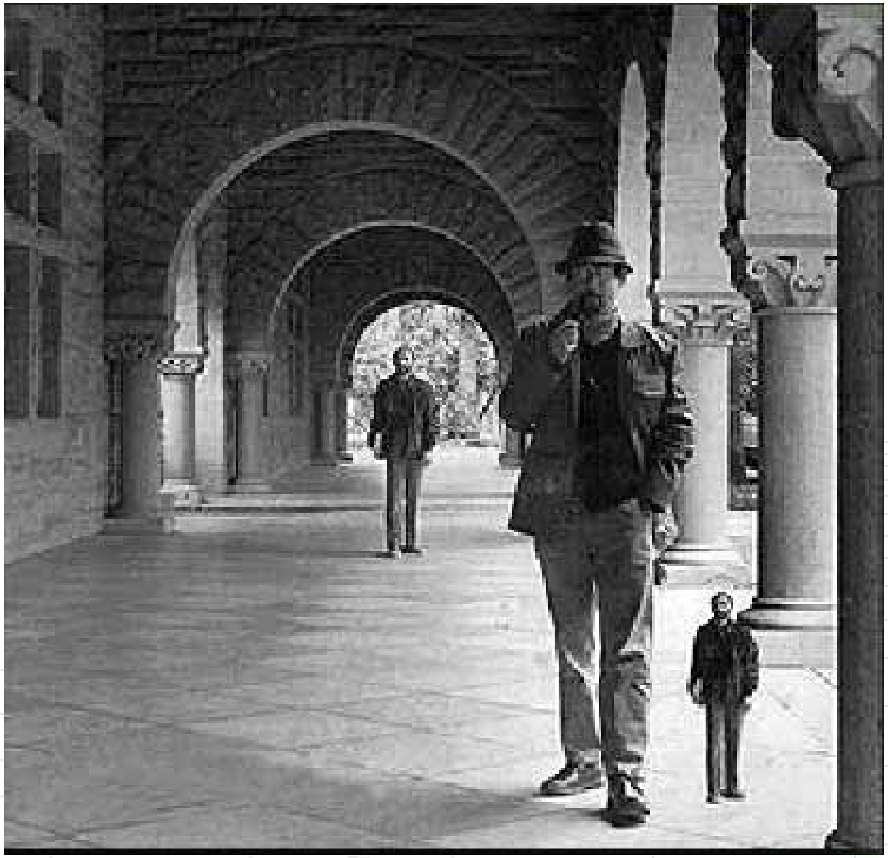
$$y^* = \frac{f}{p_z} p_y$$

$$x^* = \frac{f}{p_z} p_x$$



Perspective Projection

- What does prospective projection gives us?
  - Depth perception - objects that are far away appear smaller



Perspective Projection Properties

- Not a linear transform
- Important properties
  - Lines are preserved
  - Distances along the lines are not
  - Parallel lines are not preserved (vanishing point)

Perspective Projection Properties

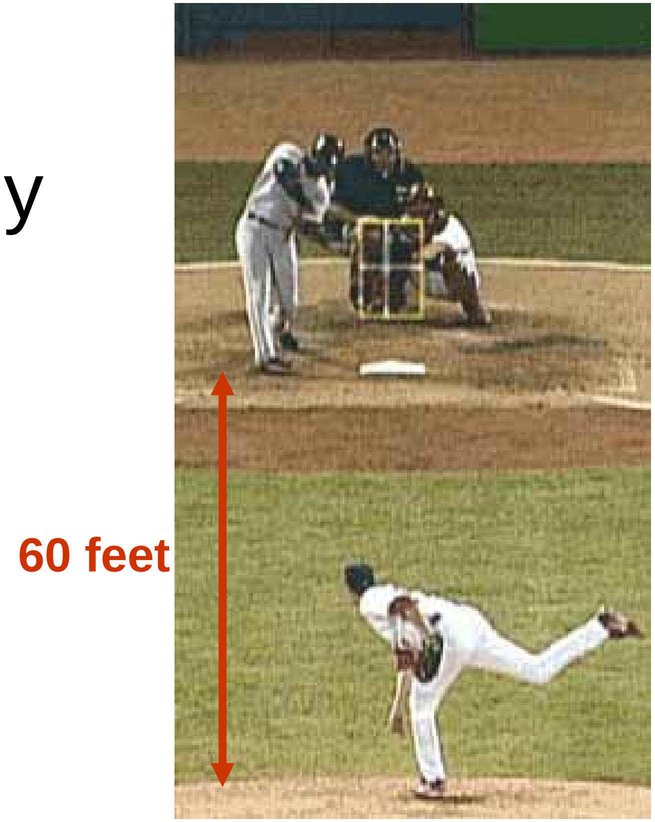
- Not a linear transform
- Important properties
  - Lines are preserved
  - Distances along the lines are not
  - Parallel lines are not preserved (vanishing point)

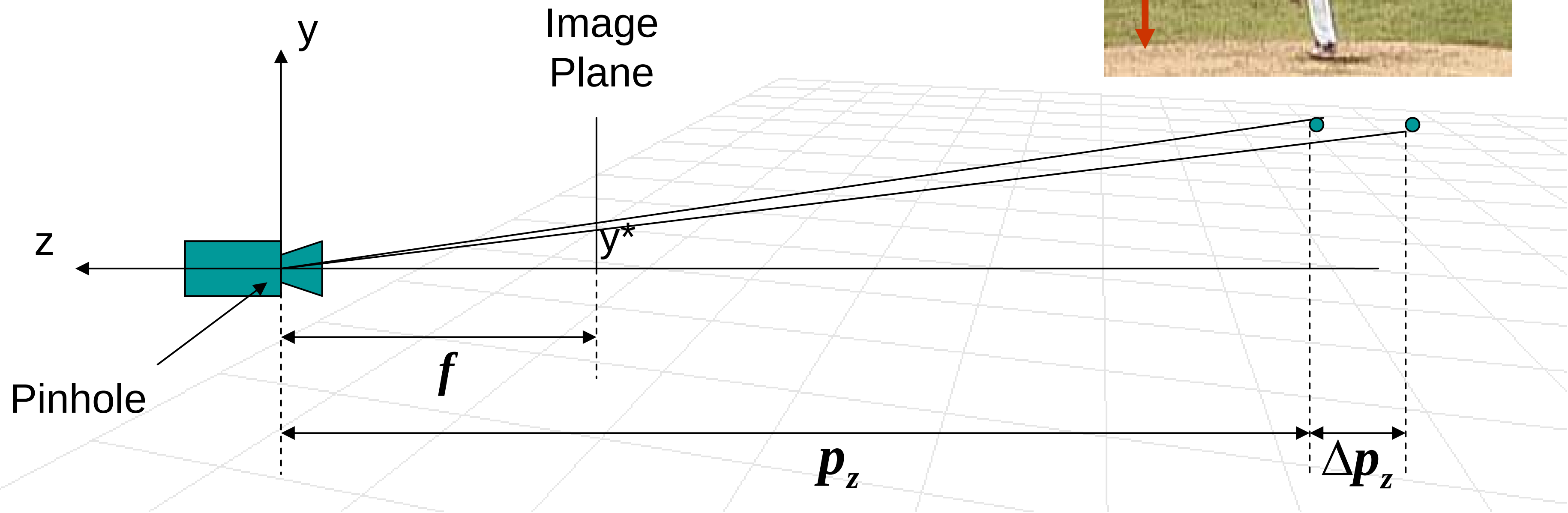
Perspective Projection Properties

- Not a linear transform
- Important properties
  - Lines are preserved
  - Distances along the lines are not
  - Parallel lines are not preserved (vanishing point)

Orthographic Projection

- What if objects are sufficiently far away?
  - Rays almost perpendicular
  - Variation in p_z is insignificant
  - For both points $y^* \approx \alpha p_y$





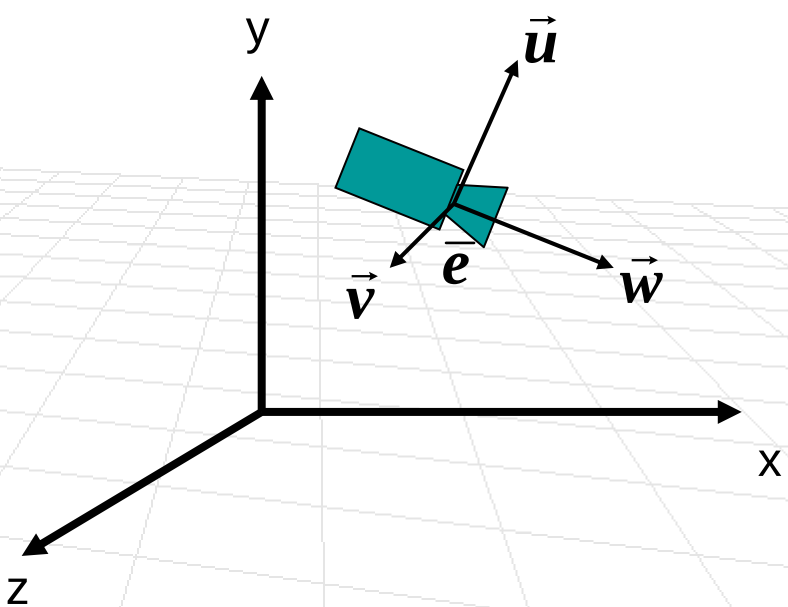
What do we really need to model to render the scene?

- Scene with 3D objects
- Position and orientation of camera in the world coordinates
- Transformation of objects from world to camera coordinates
- Project the objects onto film
- Visibility (with respect to the view volume)
  - No need to render everything, only things we can see

Position and Orientation of Camera

- In general
  - Camera can be anywhere in the world
  - Camera can move as a function of time
- How can we specify a camera coordinate frame
  - We need an origin (at the pinhole) – lets call it $\bar{e}$, and 3 unit vectors to define the camera coordinate frame $\vec{u}, \vec{v}, \vec{w}$

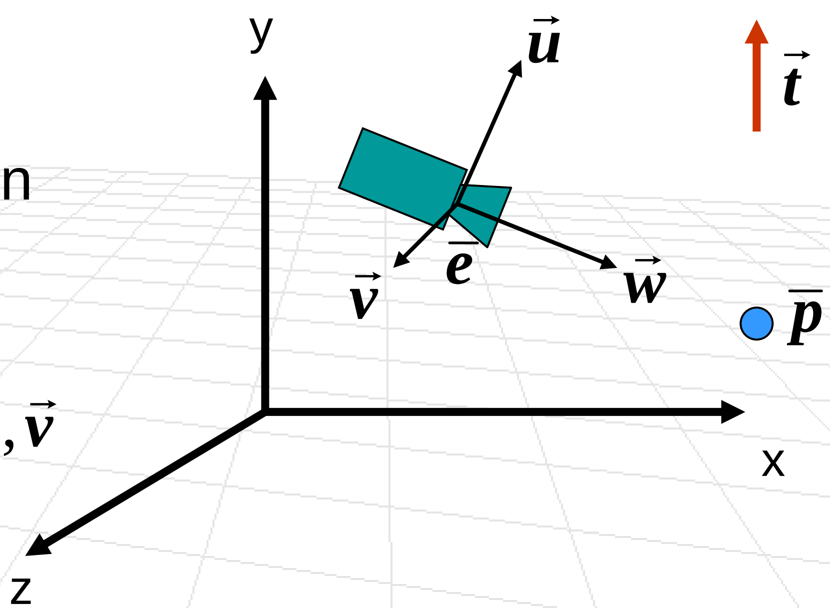
Bullet Time effect – Movie "The Matrix"



Position and Orientation of Camera

- How can we specify a camera coordinate frame
  - We need an origin (at the pinhole) – lets call it $\bar{e}$, and 3 unit vectors to define the camera coordinate frame $\vec{u}, \vec{v}, \vec{w}$
- How can we intuitively specify $\vec{u}, \vec{v}, \vec{w}$
  - Let's pick a point in the scene where we want to look, $\bar{p}$, then

$$\vec{w} = \frac{\bar{p} - \bar{e}}{\left\| \bar{p} - \bar{e} \right\|}$$



  - Designate up direction $\vec{t}$, then

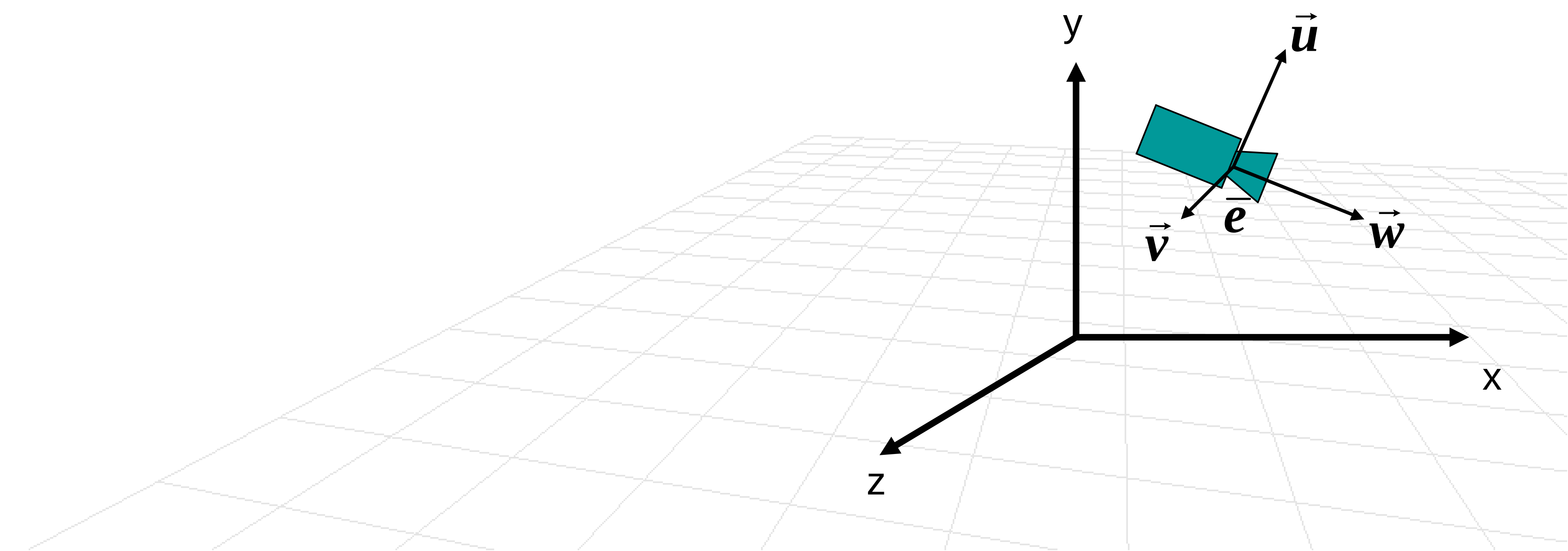
$$\vec{u} = \frac{\vec{t} \times \vec{w}}{\left\| \vec{t} \times \vec{w} \right\|}$$

  - ***v*** must be perpendicular to $\vec{u}, \vec{v}$

$$\vec{v} = \vec{w} \times \vec{u}$$

Position and Orientation of Camera

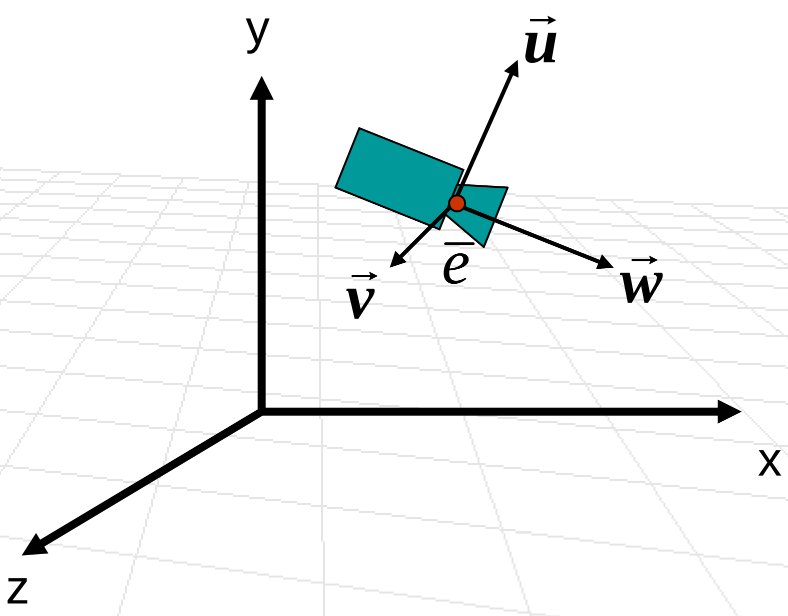
- Now that we have a camera defined in world coordinate frame, how do we take a point in the camera coordinate frame and map to the world coordinate frame?



Camera to World Transformation

- Now that we have a camera defined in world coordinate frame, how do we take a point in the camera coordinate frame and map to the world coordinate frame?
- Let's try some points

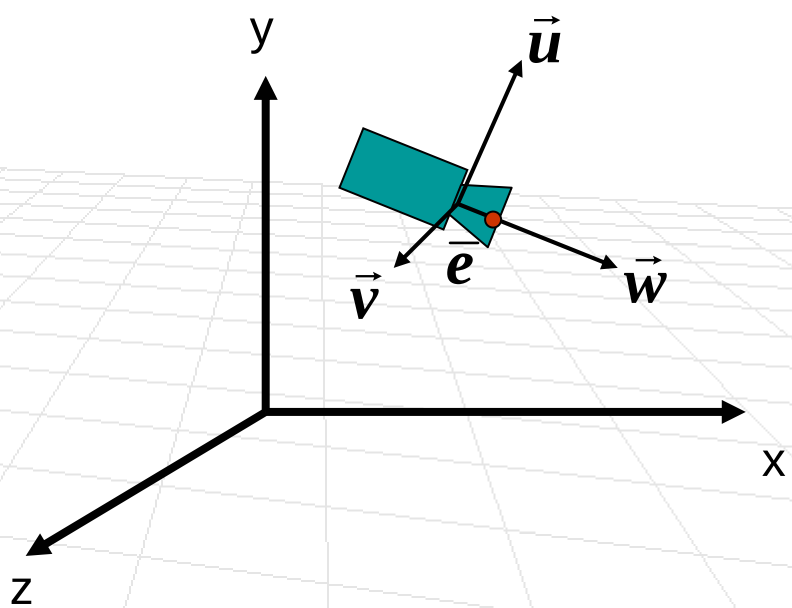
Camera Coordinates	World Coordinates
$(0,0,0)$	



Camera to World Transformation

- Now that we have a camera defined in world coordinate frame, how do we take a point in the camera coordinate frame and map to the world coordinate frame?
- Let's try some points

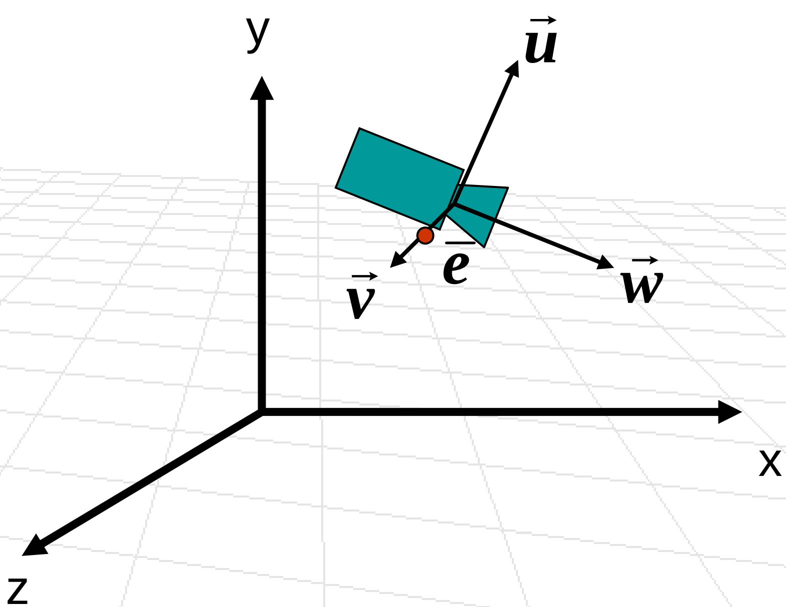
Camera Coordinates	World Coordinates
$(0,0,0)$	$\bar{e}$
$(0,0,f)$	



Camera to World Transformation

- Now that we have a camera defined in world coordinate frame, how do we take a point in the camera coordinate frame and map to the world coordinate frame?
- Let's try some points

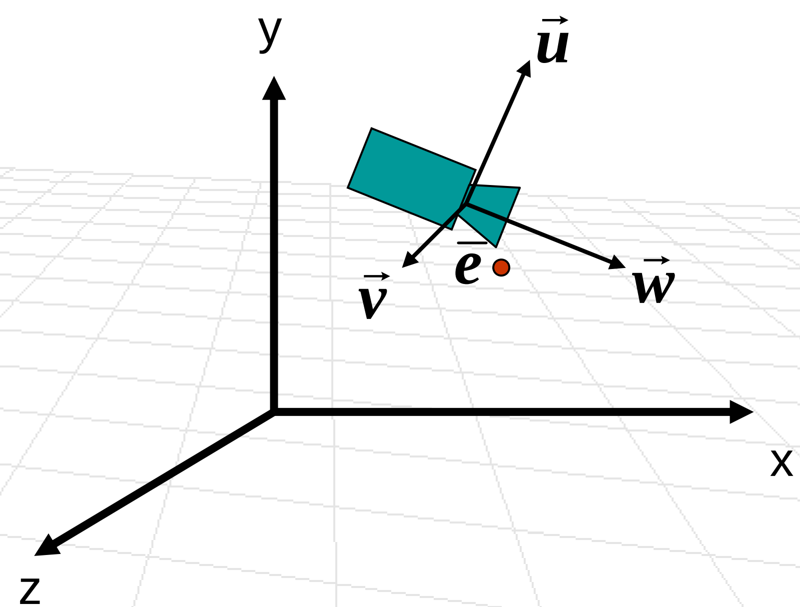
Camera Coordinates	World Coordinates
$(0,0,0)$	$\bar{e}$
$(0,0,f)$	$\bar{e} + f\vec{w}$
$(0,1,0)$	



Camera to World Transformation

- Now that we have a camera defined in world coordinate frame, how do we take a point in the camera coordinate frame and map to the world coordinate frame?
- Let's try some points

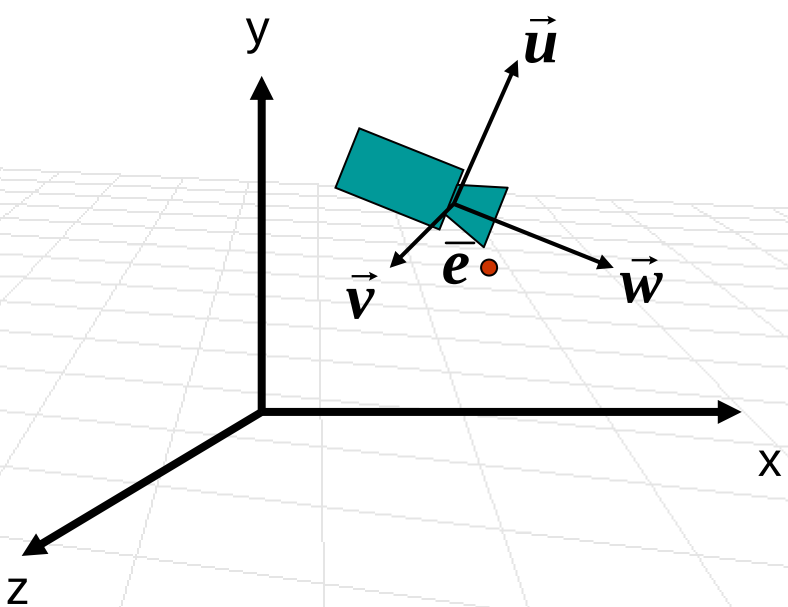
Camera Coordinates	World Coordinates
$(0,0,0)$	$\bar{e}$
$(0,0,f)$	$\bar{e} + f\vec{w}$
$(0,1,0)$	$\bar{e} + \vec{v}$
$(0,1,f)$	



Camera to World Transformation

- Now that we have a camera defined in world coordinate frame, how do we take a point in the camera coordinate frame and map to the world coordinate frame?
- Let's try some points

Camera Coordinates	World Coordinates
$(0,0,0)$	$\bar{e}$
$(0,0,f)$	$\bar{e} + f\vec{w}$
$(0,1,0)$	$\bar{e} + \vec{v}$
$(0,1,f)$	$\bar{e} + \vec{v} + f\vec{w}$



Camera to World Transformation

- It's relatively easy to show that any point in camera coordinate frame can be expressed in world coordinate frame using the following homogenized transformation:

$$\bar{\boldsymbol{p}}^{w} = \boldsymbol{M}_{cw}\bar{\boldsymbol{p}}^{c}$$

$$\boldsymbol{M}_{cw} = \begin{bmatrix} [\vec{\boldsymbol{u}}, \vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}] & \bar{\boldsymbol{e}} \\ [0,0,0] & 1 \end{bmatrix}$$

- See lecture notes for details

Camera to World Transformation

- It's relatively easy to show that any point in camera coordinate frame can be expressed in world coordinate frame using the following homogenized transformation:

$$\bar{\boldsymbol{p}}^w = \boldsymbol{M}_{cw}\bar{\boldsymbol{p}}^c$$

- Actually, what we need is the inverse:

$$\bar{\boldsymbol{p}}^c = \boldsymbol{M}_{wc}\bar{\boldsymbol{p}}^w$$

Inverting the Camera to World Transformation

We have:

$$\bar{\boldsymbol{p}}^{w} = \boldsymbol{M}_{cw}\bar{\boldsymbol{p}}^{c} \qquad \boldsymbol{M}_{cw} = \begin{bmatrix} \boldsymbol{A} & \bar{\boldsymbol{e}} \\ [0,0,0] & 1 \end{bmatrix} \qquad \boldsymbol{A} = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ \vec{\boldsymbol{u}} & \vec{\boldsymbol{v}} & \vec{\boldsymbol{w}} \\ \downarrow & \downarrow & \downarrow \end{bmatrix}$$

We want:

$$\bar{\boldsymbol{p}}^{c} = \boldsymbol{M}_{wc}\bar{\boldsymbol{p}}^{w}$$

Inverting the Camera to World Transformation

We have:

$$\overline{p}^w = M_{cw}\overline{p}^c$$

$$M_{cw} = \begin{bmatrix} A & \overline{e} \\ [0,0,0] & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ \vec{u} & \vec{v} & \vec{w} \\ \downarrow & \downarrow & \downarrow \end{bmatrix}$$

$$\overline{p}^w = A\overline{p}^c + \overline{e}$$

$$\overline{p}^c = A^{-1}\left(\overline{p}^w - \overline{e}\right)$$

We want:

$$\overline{p}^c = M_{wc}\overline{p}^w$$

Inverting the Camera to World Transformation

We have:

$$\overline{p}^w = M_{cw}\overline{p}^c \qquad M_{cw} = \begin{bmatrix} A & \overline{e} \\ [0,0,0] & 1 \end{bmatrix} \qquad A = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ \vec{u} & \vec{v} & \vec{w} \\ \downarrow & \downarrow & \downarrow \end{bmatrix}$$

$$\overline{p}^w = A\overline{p}^c + \overline{e}$$

$$\overline{p}^c = A^{-1}\left(\overline{p}^w - \overline{e}\right)$$

Since A is orthonormal (easy to check), the inverse of A is simply a transpose

$$\overline{p}^c = A^T\left(\overline{p}^w - \overline{e}\right)$$

$$\overline{p}^c = A^T\overline{p}^w - A^T\overline{e}$$

We want:

$$\overline{p}^c = M_{wc}\overline{p}^w$$

Inverting the Camera to World Transformation

We have:

$$\bar{p}^w = M_{cw}\bar{p}^c \qquad M_{cw} = \begin{bmatrix} A & \bar{e} \\ [0,0,0] & 1 \end{bmatrix} \qquad A = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ \vec{u} & \vec{v} & \vec{w} \\ \downarrow & \downarrow & \downarrow \end{bmatrix}$$

$$\bar{p}^w = A\bar{p}^c + \bar{e}$$

$$\bar{p}^c = A^{-1}\left(\bar{p}^w - \bar{e}\right)$$

Since A is orthonormal (easy to check), the inverse of A is simply a transpose

$$\bar{p}^c = A^T\left(\bar{p}^w - \bar{e}\right)$$

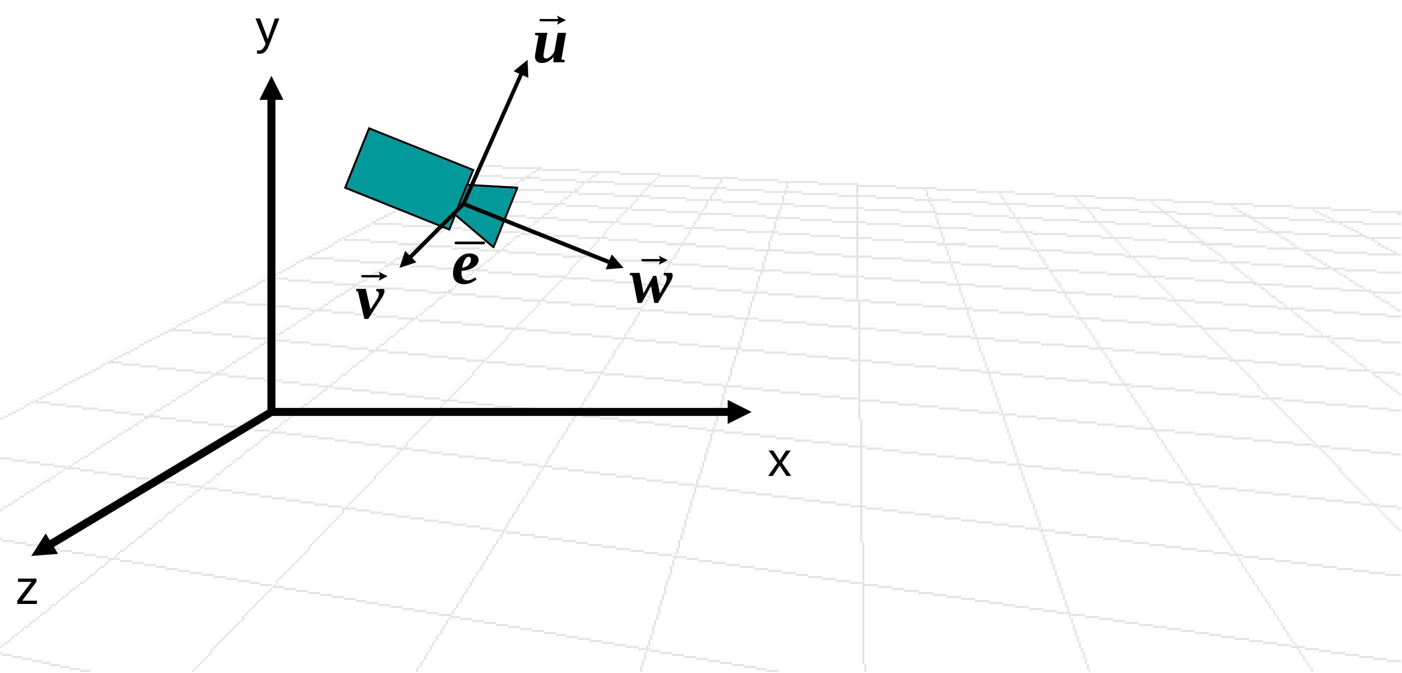
$$\bar{p}^c = A^T\bar{p}^w - A^T\bar{e}$$

We want:

$$\bar{p}^c = M_{wc}\bar{p}^w \qquad M_{wc} = \begin{bmatrix} A^T & -A^T\bar{e} \\ [0,0,0] & 1 \end{bmatrix} \qquad A^T = \begin{bmatrix} \leftarrow & \vec{u} & \rightarrow \\ \leftarrow & \vec{v} & \rightarrow \\ \leftarrow & \vec{w} & \rightarrow \end{bmatrix}$$

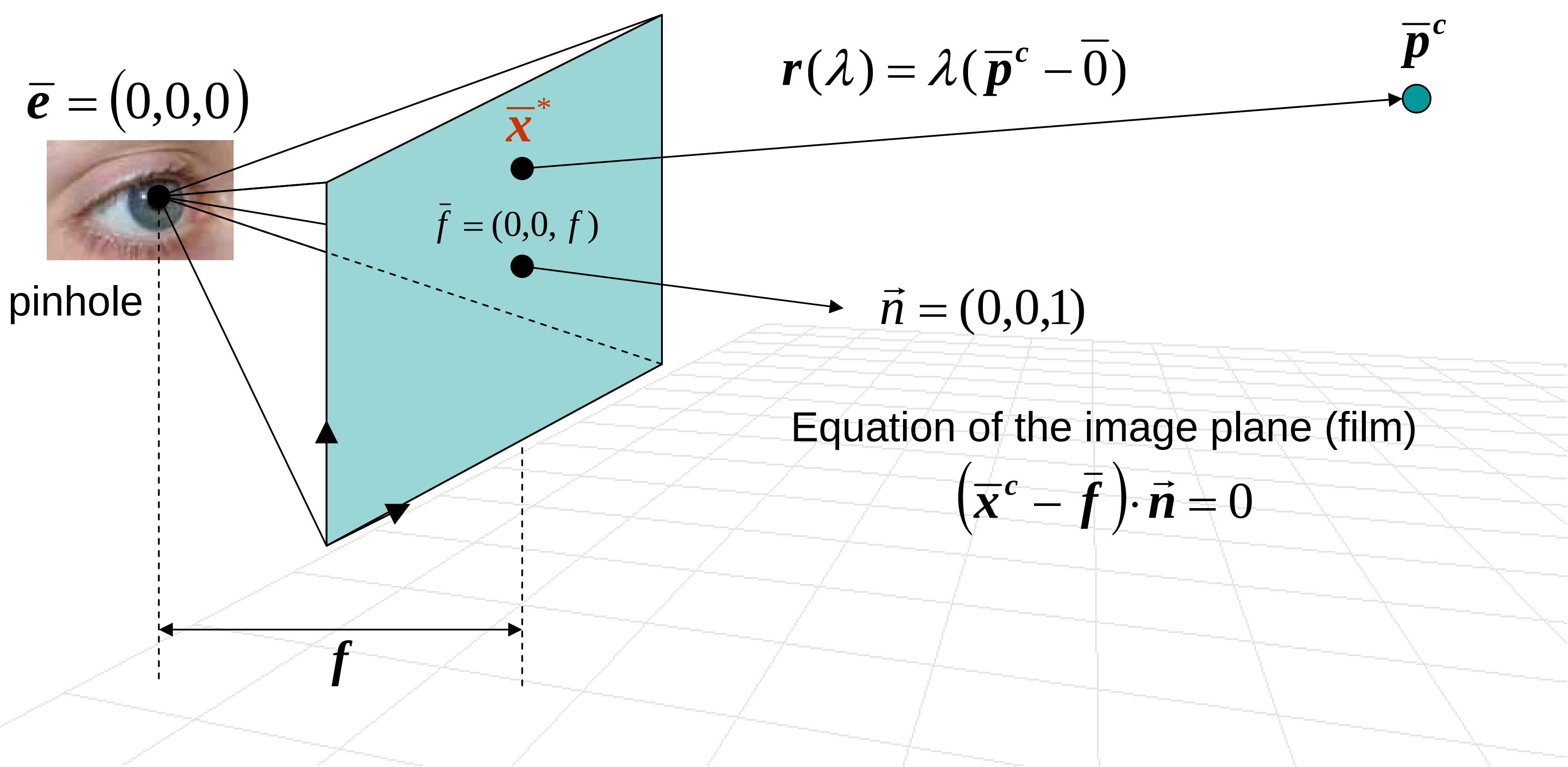
Perspective Projection (Again)

- Earlier we derive perspective projection using similar triangles
- Now, we will go through an exercise of doing it algebraically (it's a good exercise)



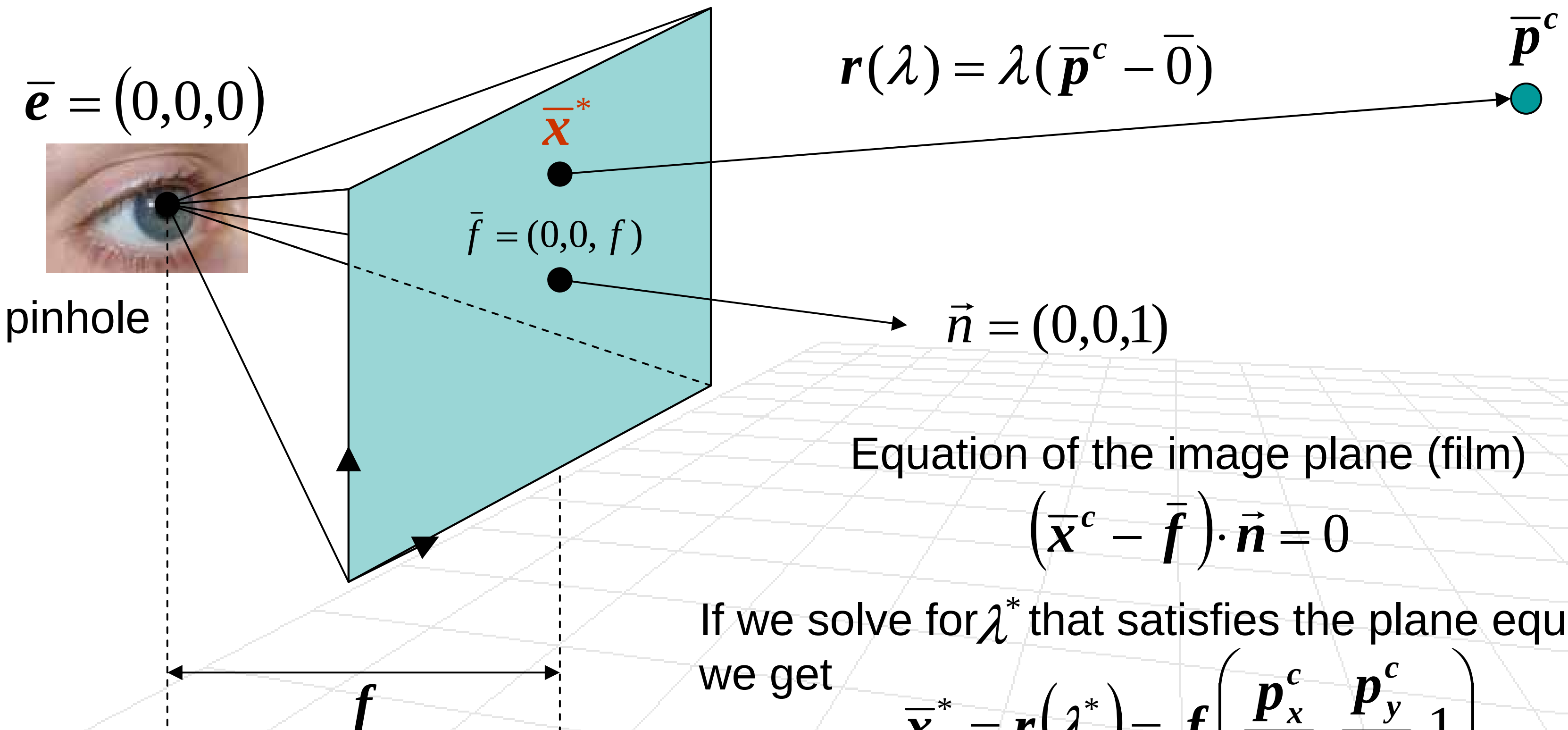
Perspective Projection

- Lets consider everything in the camera coordinate frame



Perspective Projection

- Lets consider everything in the camera coordinate frame



Equation of the image plane (film)

$$\left(\bar{x}^c - \bar{f}\right) \cdot \vec{n} = 0$$

If we solve for λ^* that satisfies the plane equation, we get

$$\bar{x}^* = r\left(\lambda^*\right) = f\left(\frac{p_x^c}{p_z^c}, \frac{p_y^c}{p_z^c}, 1\right)$$

Perspective Projection

- The mapping from a point $\bar{\boldsymbol{p}}^c$ in camera coordinates to point $\left(\boldsymbol{x}^*, \boldsymbol{y}^*, 1\right)$ in the image plane, is what we will call the **perspective projection**

$$\bar{\boldsymbol{x}}^* = \boldsymbol{r}\left(\lambda^*\right) = \boldsymbol{f}\left(\frac{\boldsymbol{p}_x^c}{\boldsymbol{p}_z^c}, \frac{\boldsymbol{p}_y^c}{\boldsymbol{p}_z^c}, 1\right)$$

Just a scaling factor, we can ignore

Homogeneous Perspective

- The mapping of point $\bar{\boldsymbol{p}}^c = \left(\boldsymbol{p}_x^c, \boldsymbol{p}_y^c, \boldsymbol{p}_z^c\right)$ to $\bar{\boldsymbol{x}}^* = \left(\boldsymbol{x}^*, \boldsymbol{y}^*, 1\right)$ is the form of scaling transformation, but since it depends on the depth of the point $\boldsymbol{p}_z^c$, it is not linear (remember the tapering example from last class)

- It would be very useful if we can express this non-linear transformation as a linear transformation (matrix). Why?

Homogeneous Perspective

- The mapping of point $\bar{\boldsymbol{p}}^c = \left(\boldsymbol{p}_x^c, \boldsymbol{p}_y^c, \boldsymbol{p}_z^c\right)$ to $\bar{\boldsymbol{x}}^* = \left(\boldsymbol{x}^*, \boldsymbol{y}^*, 1\right)$ is the form of scaling transformation, but since it depends on the depth of the point p_z^c, it is not linear (remember the tapering example from last class)

- It would be very useful if we can express this non-linear transformation as a linear transformation (matrix). Why?

$$\bar{\boldsymbol{x}}^* = \boldsymbol{M}_p \boldsymbol{M}_{wc} \bar{\boldsymbol{p}}^w$$

Homogeneous Perspective

- We can express it a a linear transformation in homogeneous coordinates (this is one of the benefits of using homogeneous coordinates!)
- Here's the transformation that does what we want:

$$\boldsymbol{M}_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/\boldsymbol{f} & 0 \end{bmatrix}$$

- Let's prove this is true

Homogeneous Perspective

- Claim:

$$\begin{bmatrix} f\begin{pmatrix} x^* \\ y^* \\ 1 \end{pmatrix} \\ 1 \end{bmatrix} = \begin{bmatrix} f\begin{pmatrix} p_x^c / p_z^c \\ p_y^c / p_z^c \\ 1 \end{pmatrix} \\ 1 \end{bmatrix} = M_p \bar{p}^c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{pmatrix} p_x^c \\ p_y^c \\ p_z^c \\ 1 \end{pmatrix}$$

Homogeneous Perspective

- Claim:

$$\begin{bmatrix} f\begin{pmatrix} x^* \\ y^* \\ 1 \end{pmatrix} \\ 1 \end{bmatrix} = \begin{bmatrix} f\begin{pmatrix} p_x^c / p_z^c \\ p_y^c / p_z^c \\ 1 \end{pmatrix} \\ 1 \end{bmatrix} = M_p \overline{p}^c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{pmatrix} p_x^c \\ p_y^c \\ p_z^c \\ 1 \end{pmatrix}$$

- Proof:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} p_x^c \\ p_y^c \\ p_z^c \\ 1 \end{bmatrix} = \begin{bmatrix} p_x^c \\ p_y^c \\ p_z^c \\ p_z^c / f \end{bmatrix}$$

Point in homogeneous coordinates can be scaled arbitrarily

Homogeneous Perspective

- Claim:

$$\begin{bmatrix} f\begin{pmatrix} x^* \\ y^* \\ 1 \end{pmatrix} \\ 1 \end{bmatrix} = \begin{bmatrix} f\begin{pmatrix} p_x^c / p_z^c \\ p_y^c / p_z^c \\ 1 \end{pmatrix} \\ 1 \end{bmatrix} = M_p \overline{p}^c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{pmatrix} p_x^c \\ p_y^c \\ p_z^c \\ 1 \end{pmatrix}$$

- Proof:

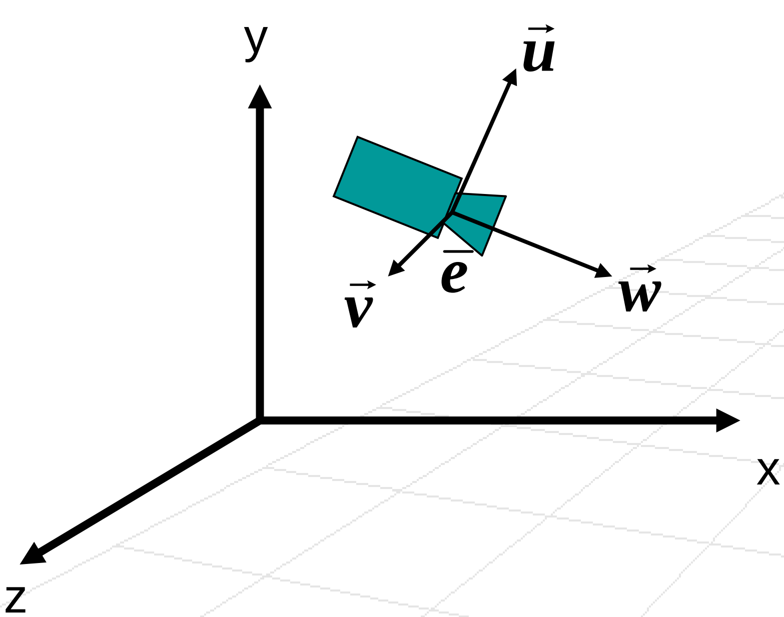
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} p_x^c \\ p_y^c \\ p_z^c \\ 1 \end{bmatrix} = \begin{bmatrix} p_x^c \\ p_y^c \\ p_z^c \\ p_z^c / f \end{bmatrix} = p_z^c / f \begin{bmatrix} f p_x^c / p_z^c \\ f p_y^c / p_z^c \\ f \\ 1 \end{bmatrix}$$

Point in homogeneous coordinates can be scaled arbitrarily

Putting together a camera model

- Projecting a world point to image (film) plane

$$\bar{x}^* = M_p M_{wc} \bar{p}^w$$



$$\bar{x}^* = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} A^T & -A^T\bar{e} \\ [0,0,0] & 1 \end{bmatrix} \bar{p}^w$$

where $A^T = \begin{bmatrix} \leftarrow & \vec{u} & \rightarrow \\ \leftarrow & \vec{v} & \rightarrow \\ \leftarrow & \vec{w} & \rightarrow \end{bmatrix}$

Pseudodepth

- We would like to change the projection transform so that z-component of the projection gives us useful information (not just a constant f)
- We want it to encode something about depth of a point. Why?